Problema 1. Escribir un programa para calcular el perímetro de una circunferencia siguiendo el diálogo:

```
radio?__
perímetro=N°
```

Solución 1. Con programa ad-hoc

```
class Programa{
static public void main(String[]args)
throws IOException
{
  double r=U.readDouble("radio?");
  U.println("perímetro=" + 2*Math.PI*r);
}
}
```

Solución 2. Con clase para circunferencias

```
class Circunferencia{
private double r;
public Circunferencia(double x){
  if(x<=0) U.abortar("radio<=0");
  r=x;
}
public double perimetro(){
  return 2*Math.PI*r;
}
}
class Programa{
static public void main(String[]args)
throws IOException{
  Circunferencia c=new Circunferencia(
                        U.readDouble("radio?"));
  U.println("perímetro=" + c.perimetro());
}
}
```

Problema 2. Calcular el área y el perímetro de un círculo

Solución 1. Modificar clase Circunferencia agregando método área

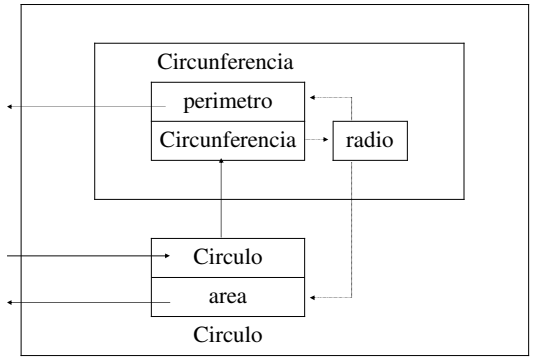
Solución 2. Usar (y escribir) nueva clase Circulo que:

- extienda la clase Circunferencia con método área
- herede las otras componentes (datos y métodos).

```
U.println("Calcular área y perímetro de círculo");

Circulo c=new Circulo(U.readDouble("radio?"));

//uso de método de clase Circulo
U.println("área="+c.area());

//uso de método heredado de clase Circunferencia
U.println("perímetro="+c.perimetro());
```

```
//clase base o superior
class Circunferencia{
protected double r;//visible en clase y extensiones
public Circunferencia(double x){
  r=x; if(r<=0) U.abortar("radio<=0");
}
public double perimetro(){
  return 2*Math.PI*r;
}
}
//clase extendida o derivada
class Circulo extends Circunferencia{
public double area(){
  return Math.PI*r*r;
}
public Circulo(double x){//constructor
  super(x);//invocar ctor de clase superior
}
}
```

class Circulo extends Circunferencia



Problema 3. Calcular área y perímetro de un círculo o un cuadrado

```
Circulo(1) o Cuadrado(2)? __
radio?__  o lado?__
area=N°
perimetro=N°
```

Programa:

```
int n=U.readInt("Circulo(1) o Cuadrado(2)?");
if(n==1){
  Circulo c=new Circulo(U.readDouble("radio?"));
  U.println("area="+c.area());
  U.println("perimetro="+c.perimetro());
}else if(n==2){
  Cuadrado c=new Cuadrado(U.readDouble("lado?"));
  U.println("area="+c.area());
  U.println("perimetro="+c.perimetro());
}else
  U.abortar("debe ser 1 o 2");
```

Solución 1. Con clases independientes

```
class Circulo{
protected double r;
public Circulo(double x){
  r=x; if(r<=0) U.abortar("radio<=0");
}
public double area(){return Math.PI*r*r;}
public double perimetro(){return 2*Math.PI*r;}
}
class Cuadrado{
protected double a;
public Cuadrado x){
  a=x; if(a<=0) U.abortar("lado<=0");
}
public double area(){return a*a;}
public double perimetro(){return 4*a;}
}
```

Solución 2. Con jerarquía de clases

```
class Figura{
protected double x;
public Figura(double x){
  this.x=x; if(x<=0) U.abortar("<=0");
}
}
class Circulo extends Figura{
public Circulo(double x){super(x);}
public double area(){return Math.PI*x*x;}
public double perimetro(){return 2*Math.PI*x;}
}
class Cuadrado extends Figura{
public Cuadrado(double x){super(x);}
public double area(){return x*x;}
public double perimetro(){return 4*x;}
}
```

Solución 3. Con métodos ficticios (redefinidos en extensiones)

```
class Figura{
protected double x;
public Figura(double x){
  this.x=x; if(x<=0) U.abortar("<=0");
}
public double area(){return 0;}
public double perimetro(){return 0;}
}
```

Corolario. Programa usuario más breve

```
Figura f;
switch( U.readInt("Circulo(1) o Cuadrado(2)?") ){
case 1: f=new Circulo(U.readDouble("radio?")); break;
case 2: f=new Cuadrado(U.readDouble("lado?")); break;
default: U.abortar("1 o 2");
}
U.println("area="+f.area());
U.println("perimetro="+f.perimetro());
```

Enlace Dinámico (dynamic binding)

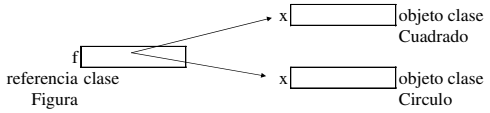
`f=new Circulo(…);  f=new Cuadrado(…);`

- referencia a objeto de clase Figura puede apuntar también a objetos de clase extendidas Circulo y Cuadrado
- objeto de clase extendida **es un** objeto de clase base (círculos y cuadrados son figuras)

`f.area()`

si f apunta a un objeto de clase

- Cuadrado, se invoca método area de clase Cuadrado
- Circulo , se invoca método area de clase Circulo



Solución 4: Con clase abstracta que obliga a redefinir métodos

```
abstract class Figura{
protected double x;
public Figura(double x){
  this.x=x; if(x<=0) U.abortar("debe ser >0");
}
abstract public double area();
abstract public double perimetro();
}
```

Notas

- no permite crear objetos, no admite new Figura()
- debe tener al menos un método abstracto:
  abstract encabezamiento;
- obliga a clases extendidas a redefinir métodos abstractos
- permite definir otras figuras

```
class Rectangulo extends Figura{
protected double y;
public Rectangulo(double x,double y){
  super(x); this.y=y; if(y<=0) U.abortar("<=0");
}
public double area(){return x*y;}
public double perimetro(){return 2*(x+y);}
}
class Triangulo extends Figura{
protected double y,z;
public Triangulo(double x,double y,double z){
  super(x); this.y=y; this.z=z;
  if(y<=0 || z<=0 || x+y<=z || x+z<=y || y+z<=x)
    U.abortar("no forman triangulo");
}
public double perimetro(){return x+y+z;}
public double area(){
  double s=(x+y+z)/2;
  return Math.sqrt(s*(s-x)*(s-y)*(s-z));
}}
```

Problema . Escriba la clase abstracta Cuerpo y las clases extendidas Cubo, Esfera y Caja para el siguiente programa que permite calcular el volumen y el área de un cubo, una esfera, o una caja (paralelepípedo)

```
Cuerpo c;
switch(U.readInt("Cubo(1),Esfera(2),o Caja(3)?"))
{
case 1: c=new Cubo(U.readDouble("lado?"));
        break;
case 2: c=new Esfera(U.readDouble("radio?"));
        break;
case 3: c=new Caja(U.readDouble("largo?"),
                   U.readDouble("ancho?"),
                   U.readDouble("alto?"));
        break;
default: U.abortar("1, 2 o 3");
}
U.println("volumen=" + c.volumen());
U.println("area=" + c.area());
```

Nota. El área de la esfera es $4\pi r^2$ y el volumen es $4/3\pi r^3$

```
abstract class Cuerpo{
protected double x;
public Cuerpo(double x){
  this.x=x; if(x<=0) U.abortar("<=0");
}
abstract double area();
abstract double volumen();
}
class Esfera extends Cuerpo{
public Esfera(double x){super(x);}
public double area(){return 4*Math.PI*x*x;}
public double volumen(){return Math.PI*x*x*x*4/3;}
}
class Cubo extends Cuerpo{
public Cubo(double x){super(x);}
public double area(){return 6*x*x;}
public double volumen(){return x*x*x;}
}
```

```
class Caja extends Cuerpo{
protected double y, z;
public Caja(double x,double y,double z){
  super(x); this.y=y; this.z=z;
  if(y<=0 || z<=0) U.abortar("<=0");
}
public double area(){
  return 2*(x*y + x*z + y*z);
}
public double volumen(){
  return x*y*z;
}
}
```

Nota. Un cubo es también una caja

```
class Cubo extends Caja{
public Cubo(double x){super(x,x,x);}
}
```

Solución 2. Con interface

- clase base abstracta
- sólo con métodos implicitamente abstractos
- sin datos

```
interface Cuerpo
{
public double area();
public double volumen();
}
```

```
class Esfera implements Cuerpo{
protected double r;
public Esfera(double x){r=x;}
public double area(){
  return 4*Math.PI*r*r;
}
public double volumen(){
  return Math.PI*r*r*r*4/3;
}
}
class Cubo implements Cuerpo{
protected double a;
public Cubo(double x){a=x;}
public double area(){return 6*a*a;}
public double volumen(){return a*a*a;}
}
```

```
class Caja implements Cuerpo{
protected double l,a,h;//largo, ancho y altura
public Caja(double x,double y,double z){
  l=x; a=y; h=z;
  if(x<=0 || y<=0 || z<=0) U.abortar("<=0");
}
public double area(){
  return 2*(l*a + l*h + a*h);
}
public double volumen(){
  return l*a*h;
}
}
```

Nota. Un cubo es también una caja

```
class Cubo extends Caja{
public Cubo(double x){super(x,x,x);}
}
```